

AD-774 452

SPEECH UNDERSTANDING SYSTEMS

James W. Forgie

Massachusetts Institute of Technology

Prepared for:

Electronic Systems Division  
Advanced Research Projects Agency

15 January 1974

DISTRIBUTED BY:

**NTIS**

National Technical Information Service  
U. S. DEPARTMENT OF COMMERCE  
5285 Port Royal Road, Springfield Va. 22151

**BEST  
AVAILABLE COPY**

AD 774452

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)  Lincoln Laboratory, M. I. T.		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE  Speech Understanding Systems			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Semiannual Technical Summary, 1 June through 30 November 1973			
5. AUTHOR(S) (Last name, first name, initial)  Forgie, James W.			
6. REPORT DATE 30 November 1973		7a. TOTAL NO. OF PAGES 32	7b. NO. OF REFS 5
8a. CONTRACT OR GRANT NO. F19628-73-C-0002		9a. ORIGINATOR'S REPORT NUMBER(S) Semiannual Technical Summary, 30 November 1973	
8b. PROJECT NO. ARPA Order 2006		9b. OTHER REPORT NUMBER (Any other numbers that may be assigned this report) ESD-TR-73-340	
10. AVAILABILITY/LIMITATION NOTICES  Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES  None		12. SPONSORING MILITARY ACTIVITY  Advanced Research Projects Agency, Department of Defense	
13. ABSTRACT  Work has shifted from research in the areas of phonetic recognition and linguistic processing to integration of an experimental speech understanding system. This mid-term system, scheduled for completion in April 1974, will be composed of phonetic recognition, linguistic processing, and functional response modules. The system task is the vocal command of a data retrieval, analysis, and display facility intended to support a researcher in studying the acoustic correlates of phonemic events.  A first version of an acoustic phonetic "front end" has been developed and tested with a corpus of 73 sentences. It has been operated in the system environment with two different linguistic processing modules, and encouraging results have been achieved in preliminary testing. The tests involved vocabularies of 200 to 500 words and context-free grammars of 111 to 300 production statements.  The complete mid-term system will include a functional response module and a different grammar from those being used currently. The new grammar and functional response module have been specified.  The speech data base now contains 185 utterances with phonetic labels and acoustic processing results. The SURNET server which provides network access to the data base has been converted to use the network file transfer protocol.  New magnetic-tape and disk systems have been added to the TX-2 system. The new disk now handles all file storage and user swapping with significant improvement in system performance.			
14. KEY WORDS  speech understanding systems linear predictive coding phonetic recognition VASSAL  CASPERS SURNET LPARS TX-2 system			

Reproduced by  
NATIONAL TECHNICAL  
INFORMATION SERVICE  
U S Department of Commerce  
Springfield VA 22151

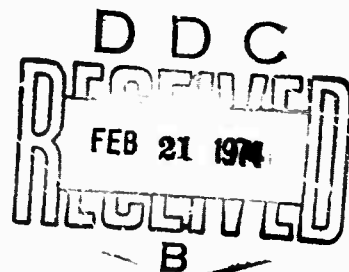
MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
LINCOLN LABORATORY

SPEECH UNDERSTANDING SYSTEMS

SEMIANNUAL TECHNICAL SUMMARY REPORT  
TO THE  
ADVANCED RESEARCH PROJECTS AGENCY

1 JUNE - 30 NOVEMBER 1973

ISSUED 15 JANUARY 1974



Approved for public release; distribution unlimited.

LEXINGTON

MASSACHUSETTS

ja

## SUMMARY

Work has shifted from research in the areas of phonetic recognition and linguistic processing to integration of an experimental speech understanding system. This mid-term system, scheduled for completion in April 1974, will be composed of phonetic recognition, linguistic processing, and functional response modules. The system task is the vocal command of a data retrieval, analysis, and display facility intended to support a researcher in studying the acoustic correlates of phonemic events.

A first version of an acoustic phonetic "front end" has been developed and tested with a corpus of 73 sentences. It has been operated in the system environment with two different linguistic processing modules, and encouraging results have been achieved in preliminary testing. The tests involved vocabularies of 200 to 500 words and context-free grammars of 111 to 300 production statements.

The complete mid-term system will include a functional response module and a different grammar from those being used currently. The new grammar and functional response module have been specified.

The speech data base now contains 185 utterances with phonetic labels and acoustic processing results. The SURNET server which provides network access to the data base has been converted to use the network file transfer protocol.

New magnetic-tape and disk systems have been added to the TX-2 system. The new disk now handles all file storage and user swapping with significant improvement in system performance.

Accepted for the Air Force  
Eugene C. Raabe, Lt. Col., USAF  
Chief, ESD Lincoln Laboratory Project Office

## CONTENTS

Summary	iii
Glossary	vii
 I. THE LINCOLN MID-TERM SYSTEM	 1
II. PHONETIC RECOGNITION	3
III. LINGUISTICS	11
A. VASSAL	11
B. CASPERS	12
C. General Syntax Program (GSP)	15
IV. MID-TERM SYSTEM TASK	16
A. Subtasks	17
B. User Model	18
C. Grammar	19
D. Functional Response	19
V. SPEECH DATA BASE	20
A. Data Base Software	20
B. SURNET	21
VI. SYSTEM ACTIVITIES	21
A. TX-2 System Hardware	21
B. TX-2 System Software	22
C. TSP System	23
References	23

## GLOSSARY

APEL	Acoustic phonetic element – output from the acoustic phonetic front end
APEX	TX-2 time-sharing system
ARPA	Advanced Research Projects Agency
CASPERS	A system which processes an APEL string to find and score sentences
FDP	Fast Digital Processor – Lincoln Laboratory computer designed for waveform processing applications
GSF	General Syntax Processor – a software system to support natural language processing
SATS	Semiannual Technical Summary Report
SURNET	Specialized server process intended to provide network access to speech data base on TX-2
TSP	Terminal Support Processor
VASSAL	A system which processes an APEL string to find and score sentences

## SPEECH UNDERSTANDING SYSTEMS

### I. THE LINCOLN MID-TERM SYSTEM

During the current reporting period, Lincoln Laboratory work in speech understanding systems has shifted from research and experimentation in the areas of phonetic recognition and linguistic processing, to the integration of experimental program modules into an experimental speech understanding system. This mid-term system is intended to serve as a test bed for evaluating techniques already under development, and to focus continuing research into remaining problem areas.

The task of the mid-term system is the vocal command of a data retrieval, analysis, and display facility intended to support a researcher in studying the acoustic correlates of phonemic events. The task makes use of the Lincoln speech data base and display facilities available on TX-2. The task is goal-oriented with some predictable sequential behavior. The task as presently specified requires a vocabulary of about 200 words and is described by a finite context-free grammar. The domain can be readily extended to larger vocabularies and more general grammars as needed to test the effectiveness of the algorithms being explored in the system.

The design of the mid-term system stresses modularity and assumes a relatively straightforward flow of information and control from speech input to action output. Figure 1 is a schematic representation of the system showing its major components and communication paths. Successful "understanding" of a spoken command or query is to be taken as the execution of an appropriate action by the system.

In general terms, the functions performed by the modules shown in Fig. 1 are:

- (a) The Phonetic Recognition module performs in two modes. In its "front end" mode, it takes a speech waveform as input and produces a sequence of acoustic-phonetic elements (APELs) which can be used by the Linguistic Processing module to hypothesize words and sentences. In its verification mode, it judges the adequacy of fit between a hypothesized phonetic sequence and the actual acoustic signal produced by the speaker.
- (b) Linguistic Processing in the mid-term system involves the use of a lexicon and grammar appropriate to the system task to propose and score sentences with respect to information provided by the phonetic recognition module. In our conception, the grammar provides constraints on both the syntax and semantics of sentences in the limited task domain. Linguistic Processing, when successful, will find one or more sentence candidates which, with their underlying parse structures and scores, serve as input to the Functional Response module.
- (c) In addition to calling for overt actions such as searching the data base or presenting a display, the Functional Response module maintains a representation of the miniworld of the system task which we call a Task Model. Information from this model is used to help select or reject sentence candidates proposed by Linguistic Processing. In the event that no acceptable sentence candidate with an adequate score is found, the Functional Response module must ask the speaker to repeat or rephrase his request.

Fig. 1. Block diagram of Lincoln mid-term system.

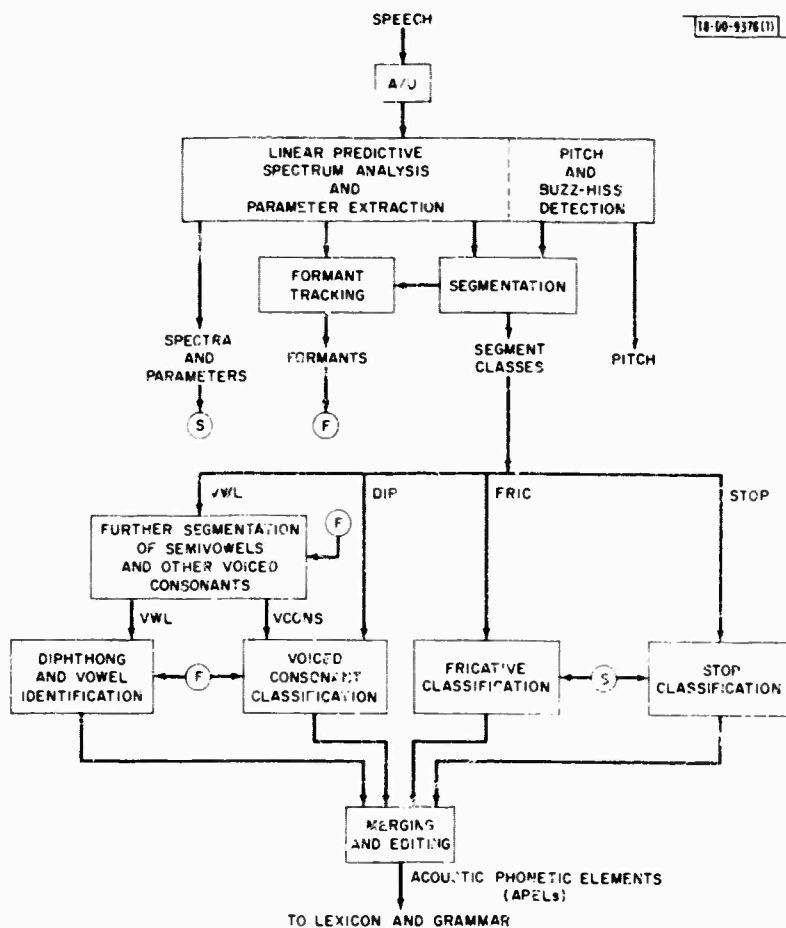


Fig. 2. Overview block diagram of acoustic phonetic front end.

The mid-term system is scheduled for demonstration in April 1974. At present, its modules are in various states of development. A first version of a phonetic recognition module has been connected to each of two alternative linguistic processing modules and operated successfully. Functional response is still in a design stage. These system elements are discussed in greater detail in Secs. II through IV.

## II. PHONETIC RECOGNITION

Work in phonetic recognition has concentrated on the development of an acoustic-phonetic "front end" making use of segmentation, formant tracking, and phoneme identification work discussed in previous SATS reports.<sup>1</sup> Previously discussed work on verification has been deferred to concentrate more effort on the front end and its integration into the mid-term system. Substantial development of the front end has taken place since the last SATS. This section describes the current state of the front end and the results of tests on a corpus of 73 sentences.

An overview block diagram of the current acoustic-phonetic front end is shown in Fig. 2. The operations indicated in the top part of the diagram, including formant tracking and segmentation, generally require much computation of a signal processing nature and are implemented on the FDP-1219 facility. The identification and classification algorithms indicated in the bottom part of the diagram are implemented on TX-2. The initial segmentation algorithm divides the waveform into the broad phonetic classes (VWL, DIP, ERIC, and STOP) which are passed onto TX-2 for more segmentation and finer classification. Formant tracking is carried out during voiced sounds, and the formant frequencies, formant frequency motions, and formant amplitudes are analyzed on TX-2 to identify the vowels and voiced consonants which make up VWL and DIP segments. Other characteristics of the spectrum are analyzed in the classification of stops and fricatives. The output of the front end is a string of phoneme-like units which we refer to as APELs.

A list of the APELs now produced, with examples to indicate their approximate phonetic significance, is shown in Fig. 3. It should be noted that some of the APELs, such as /n/ and /f/, represent phoneme classes rather than phonemes. Also, certain APELs (/n/ and /w/) are to be taken as deletable, signaling lower confidence in detection of the associated segment. In Fig. 4, an acoustic-phonetic analysis for a sample sentence is shown. Below the linear-predictive spectrogram is a hand-labeled phonemic transcription<sup>†</sup> and below that the machine-generated APELs. Note that one to three choices are given for each APEL; for example, the initial phoneme /E/ is identified correctly on the third choice. Also, phonemes sometimes are omitted in the APEL output; for example, the /t-D/ combination in "edit the" is seen as the single APEL /p/. A spurious APEL is sometimes inserted, as in the /f/ before the final stop /-/ in "phonemic." These insertions and deletions are key problems facing the linguistic analysis. In this sentence, second formant motion was used successfully to detect the initial /l/ in "label" (as the APEL /w/); and the two nasals in "phonemic" were successfully identified. Having given this overview of the acoustic front end, we will now go into some added detail on some of the individual identification algorithms and their performance.

The framework for vowel and diphthong identification is shown in Fig. 5. At the focus of this analysis is a steady-state vowel identifier, which compares F1 and F2 at the segment center

<sup>†</sup>In this report, phonemes are represented in a single-character alphabet shown in Table 1. This alphabet is based on the single-character ARPA standard representation, modified where necessary to be printable with characters available on TX-2 consoles.

### Vowels and Diphthongs

i	beat
I	bit
E	bet
→	bat
c	bab, down
u	boat
⊃	bought, boat
A	but
U	book
R	bird
x	about (back weak vowel)
X	roses (front weak vowel)
L	bottle
Y	buy
V	non-identified vowels

### Fricatives

f	fat, thing
s	sat, zaa
S	shut, azure

### Stops

+	voiced silence
-	unvoiced silence
p	pet
b	bet
k	caught
g	gane
t	ten, keen
d	debt, geezer
c	try
C	dry

### Semivowels and Other Voiced Consonants

n	met, net, sing
n	deletable n
w	wet, iet
w	deletable w
r	rent
F	better (flapped t)
Δ	vot, that, ahead

18-2-11302

Fig. 3. Acoustic phonetic elements (APELs) produced as output of acoustic phonetic front end.



Fig. 4. Sample sentence showing front-end analysis, hand-labeled phonemes, and APEL output.

TABLE I  
TX-2 SINGLE-CHARACTER PHONEME REPRESENTATION  
WITH EXAMPLE WORDS

Phoneme	Example	Transcription	Phoneme	Example	Transcription
i	beat	bit	G	sing	siG
l	bit	blt	p	pet	pEt
e	bait	bet	t	ten	tEn
E	bet	bEt	k	kit	klt
→	bat	b→t	b	bet	bEt
a	Bob	bab	d	debt	dEt
A	but	bAt	g	get	gEt
⊃	bought	b⊃t	h	hat	h→t
o	boat	bot	C	church	CRC
U	book	bUk	J	judge	JAJ
u	boot	but	f	fat	f→t
x	about	xbWt	T	thing	TIG
X	roses	rozXz	s	sat	s→t
R	bird	bRd	S	shut	SAt
W	down	dWn	v	vat	v→t
Y	buy	bY	D	that	D→t
O	boy	bO	z	zoo	zu
y	you	yu	Z	azure	→Zyur
w	wit	wlt	L	battle	b→FL
r	rent	rEnt	M	bottom	baFM
l	let	lEt	N	happen	h→PN
m	met	mEt	F	batter	b→Fxr
n	net	nEt	Q	(glottal stop) see each	siQiC

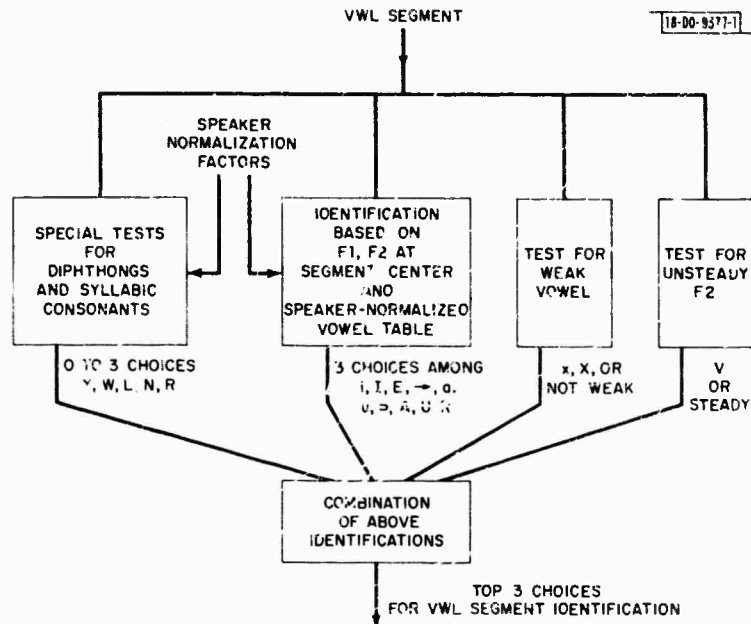


Fig. 5. Block diagram of vowel and diphthong identification processing.

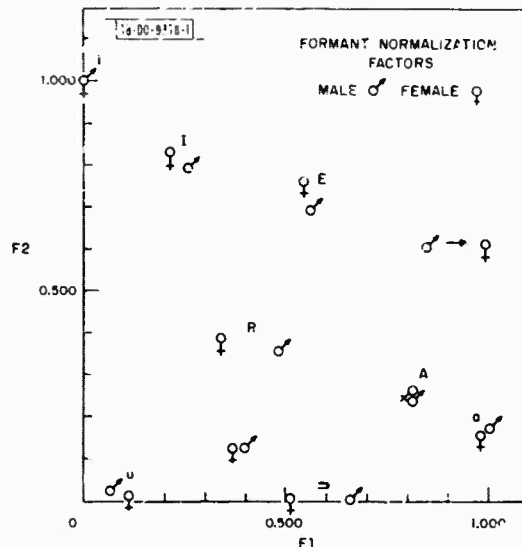


Fig. 6. Speaker normalization procedure for vowels.

To determine formant table entry for any Speaker S,  
Sex X, and Vowel V:

$$F1(V, S) = F1MIN(S) + F1X(V) F1Range(S)$$

$$F2(V, S) = F2MIN(S) + F2X(V) F2Range(S)$$

F1MIN, F1Range, F2MIN, F2Range are determined empirically for each speaker from a calibration utterance

to the corresponding formant values for a set of standard vowels for the given speaker, and generates the top three choices. But in order to enhance the performance of this steady-state vowel identification, it is quite helpful to identify separately the "non-steady" vowels such as diphthongs, and the weak (unstressed or reduced) vowels whose formant positions are quite dependent on their environment. Weak vowels are detected by energy comparisons with adjacent syllables. As examples of other special identification schemes, the /Y/ (as in "buy") is detected via its characteristic F1, F2 trajectories, and the /R/ (as in "bird") primarily through the behavior of F3. The procedure used to generate a formant table for a given speaker is indicated in Fig. 6. The speaker normalization factors were obtained by normalization of the Peterson-Barney data.<sup>2,3</sup> Note that only four formant values are necessary to determine the entire table for a speaker; these values could be obtained from a single calibration sentence. A confusion matrix indicating the performance of the steady-state vowel identification is given in Fig. 7. The data were collected on 73 sentences, divided among 3 speakers. Only the first choice APEL was entered in the matrix, and some rows and columns were combined for better readability. The number of "false detections" of weak (/x, X/) and neutral (/A, U/) vowels is not surprising, since most vowels in continuous speech become neutralized or weakened unless they are stressed.

The /r/ detection algorithm, illustrated in Fig. 8, is an example of an algorithm which uses formant motion to detect and identify a particular phoneme. On top of the figure are shown typical formant trajectories for an /r-front vowel/ combination (such as in "read"). The basic property exploited by the algorithm is that during the /r/, F3 moves low and close to F2. On the basis of the five measurements indicated on the figure, a decision is made on whether or not an /r/ is present. The vowel /R/ (as in "bird") is also detected by analysis of F3 and its relation to F2. Combined data on the detection of /r/ and /R/ are given in Fig. 9. The present algorithm attempts to detect only those /r/'s which are followed by vowels. It was found that detection of post-vocalic /r/'s is substantially less reliable, in part due to the fact that many speakers tend to slur or omit such /r/'s. In the 73 sentences, about 1000 syllables were studied; more than 10 percent of these syllables contained /r/ or /R/, so this is indeed an important phoneme. The detection of /r/ and /R/ was generally reasonably good, and it is worth noting that this detection rests heavily on the fact that we have a reliable formant tracker which works well even when two formants come quite close together, as in /r/ or /R/.

The nasal detection algorithm, illustrated in Fig. 10, uses formant positions and amplitudes to identify a DIP segment as a nasal /m, n or G/. DIPs are identified as /n/ (nasal), /w/ (either "wet" or "let"), /F/ (flapped t), /r/, or /Δ/ (other) by means of a battery of such algorithms. Some confusion data on DIP identification is given in Fig. 11.

Fricative segments are categorized into three classes by means of the algorithm shown in Fig. 12. General spectral character rather than formant structure, is used to distinguish the fricatives. The decision process indicated in Fig. 12 is carried out on every 5-msec spectral section in a FRIC segment, and a majority vote is taken to determine the segment identity. A confusion matrix for fricative identification is given in Fig. 13.

The stop analysis algorithm makes a decision on each stop as to whether it is voiced or unvoiced, and aspirated or unaspirated. If aspirated, the frequency location of BURST of the major concentration of energy in the burst spectrum is found. If BURST is high, the stop is assigned the API /p/ or /d/ depending on voicing; low f BURST stops are called /k/ or /g/; and mid-frequency BURST stops are called /c/ or /C/. If the burst is a flat spectrum, the stop is called /p/ or /b/. This rudimentary algorithm gives somewhat spotty results, but is reasonably

HAND-LABELED PHONEMES	APEL S							
	i, I	E	—	R	A, U	o, ɔ	u	x, X
i, I	54	11		4	3			26
e, E	19	51	8		14			8
—		10	11		8	3		
R				26	8			7
A, U	3	3			13			10
o, ɔ, o					30	21		4
u	4					6	5	3
x, X	5	3		4	11		4	81

Fig. 7. Confusion matrix for vowel identification showing results from 73 sentences (3 speakers) scoring first choice only.

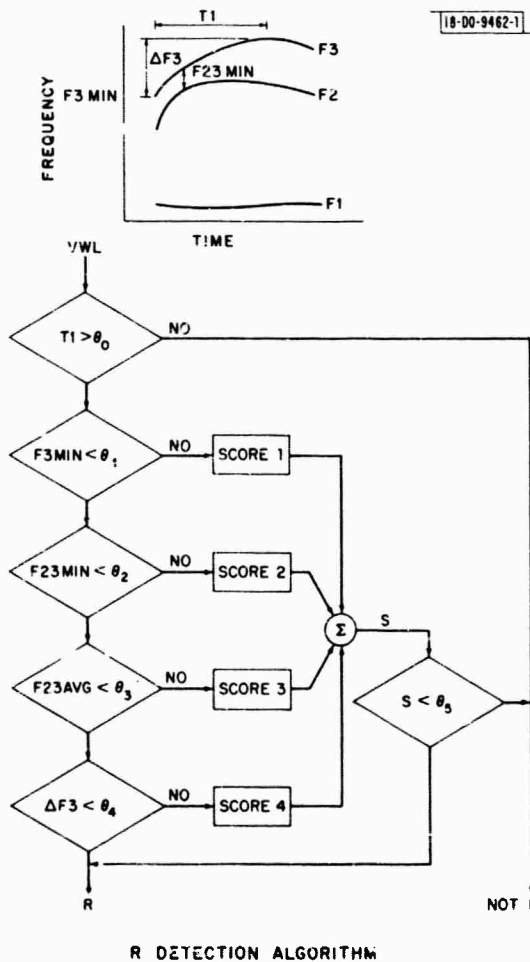


Fig. 8. Flow chart for /R/ detection algorithm.

18-2-11363

Fig. 9. Confusion matrix for /r/ and /R/ identification showing results from 73 sentences (3 speakers).

	r DETECTED	R DETECTED	NO r OR R
SYLLABLE CONTAINS PRECEDING r	44	4	8
SYLLABLE CONTAINS R	3	44	12
SYLLABLE CONTAINS NO r OR R	2	3	961

18-DO-9464-1

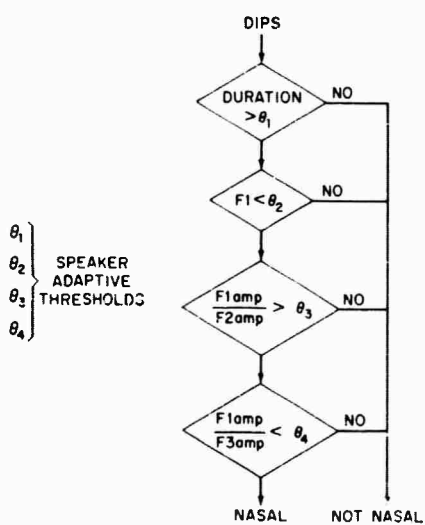


Fig. 10. Flow chart for nasal detection algorithms.

18-2-11364

HANO-LABELED PHONEMES	APELS				
	n	w	F	r	Δ
NASAL	55	7	4	3	13
w, l	1	22	1		10
FLAPPED l	1		13		4
r	2			8	3
v, O	18	1			6
VOWEL	2	1	4		13
OTHER	9	1	3	1	13

Fig. 11. Confusion matrix for DIP identification showing results from 73 sentences (3 speakers).

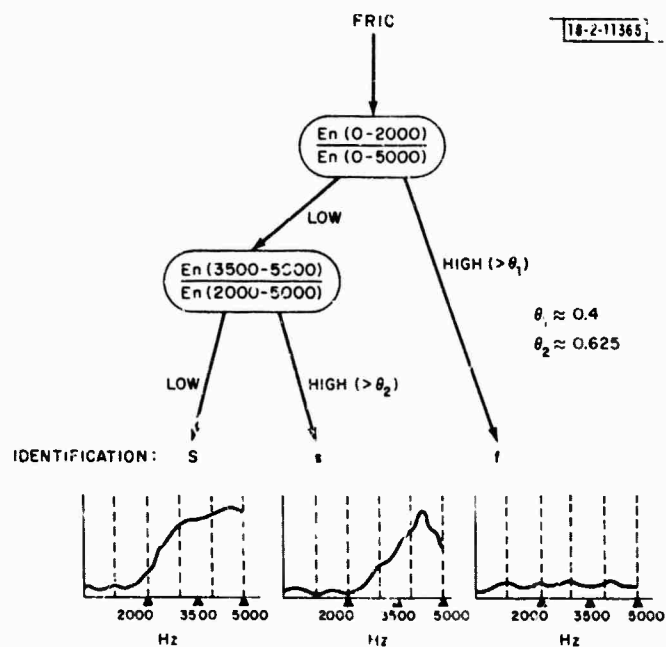


Fig. 12. Flow chart for fricative identification algorithm.

18-00-946(1),

HAND-LABELED PHONEMES	APELS		
	f	s	S
f, T	30	8	0
s, z	1	107	0
S, Z	1	2	6
h	6	2	1
STOP AND AFFRICATE	10	8	0

Fig. 13. Confusion matrix for fricative identification showing results from 73 sentences (3 speakers).

effective when a strong burst is present, as is often the case when the stop begins a stressed syllable.

A great deal of work clearly remains to be done on the front end. For example, coarticulation effects need to be exploited for better identification of consonants such as stops; indications of stress and confidence should be added to the front-end output; and much more needs to be done in the area of speaker normalization. However, we feel that the acoustic front end presented here is quite adequate for carrying out meaningful experiments and studies in the vital area of acoustic-linguistic linking.

### III. LINGUISTICS

Our work in linguistics is pursuing three alternative approaches to the problem of finding and scoring sentence candidates. The three approaches are intended to explore different parse and scoring strategies, different points of application of phonological rules, and the potential for parallel processing. All are being developed as modules for testing in the mid-term system. The approaches are identified by the names VASSAL, CASPERS, and GSP. The following three sections discuss the concepts being explored by each system and present their current status.

#### A. VASSAL

VASSAL is the name we have given to the linguistic processing module which is the outgrowth of our previously reported work on the application of heuristic search techniques to lexical segmentation and parsing. VASSAL makes use of a dictionary containing the nominal phonemic pronunciations of the words in the grammar. Semantic information is represented in the grammar by replacing syntax classes with semantic classes appropriate to the task domain. Word candidates are scored using a matrix derived from the 73 sentence corpus used to test the front-end processing.

A small set of phonological and front-end dependent rules are applied when matching APEL strings with dictionary entries. These rules attempt to account for alternative pronunciations and word boundary effects.

The emphasis in the development of VASSAL has been on the exploration of the effectiveness of various heuristics in controlling the search for acceptable sentence candidates. The current strategy is completely top-down and left-to-right in parsing a sentence, but a major development goal is the introduction of bottom-up information to be obtained by making a preliminary scan to detect the presence of easily recognized words. If any such words are to be found, an attempt would be made to develop complete grammatical substructures by parsing outward in both directions. The result should provide "islands of reliability" which could guide the overall parse past a stretch of badly garbled input which could defeat a pure left-to-right approach.

The current version of VASSAL uses a cumulative scoring strategy which causes longer paths to be preferred. The resulting strategy is not a simple "depth first" one, however, because the scoring matrix relating APELs to dictionary phonemes contains negative numbers which can cause the score of a long path to drop below that of a shorter one. In such a case, the longer path will cease to be pursued even though it is still grammatically acceptable.

In tests to date, VASSAL has used a semantic grammar of 111 production rules and a vocabulary of 248 words. The grammar is finite (nonrecursive) and will generate more than 4.8 million sentences. All the sentences are commands to our speech data retrieval, analysis, and display system. There are many similarities between this test grammar and that for the

mid-term system discussed below in Sec. IV. The task domain for the current test situation covers a broader range of activities and lacks the goal-oriented flow of the mid-term system task.

Sample sentences generated by the test grammar are:

- "Complete the average energy for the second voiced segment."
- "Get all hand labels from the tape."
- "List the unvoiced fricatives from sentence one."

Since VASSAL and the acoustic phonetic front end reached an operational status just a few days before the end of this reporting period, very little testing has been carried out as yet. One run was made with a set of 34 sentences divided among three speakers. Since the heuristics used do not guarantee that sentences will be found in order of decreasing score and since we wish to avoid an exhaustive search, a cutoff time of 5 CPU seconds was set on the test run. In the test, VASSAL found the correct sentence for about half the samples from two of the speakers. Results for the third speaker were much worse. Analysis of the failures has led to suggestions for changes and improvements in the front end, in dictionary scoring, and in the choice of heuristics.

## B. CASPERS

CASPERS is the name given to a system being developed as part of the on-going doctoral research of John W. Klovstad at M.I.T. He conceives CASPERS being capable of performing all functions of a speech understanding system beyond the acoustic phonetic front end. The system provides a general framework and set of algorithms for performing dictionary degarbling, parsing, semantic testing, and functional response control. CASPERS can best be viewed perhaps as a special-purpose programming system in which some person other than the system designer uses CASPERS to realize a speech understanding system oriented toward a particular task. In our work, we are acting as a user of CASPERS in our application of part of its capabilities to realize a linguistic module appropriate to the system concept discussed in Sec. I of this report. In this section, we describe the operation of CASPERS and discuss aspects of its design which relate to our application. Discussion of other aspects of CASPERS, such as the efficiency of its algorithms and the effectiveness of its use of available information, which have received much attention in Klovstad's doctoral research, will be deferred until the completion of the research.

Application of CASPERS proceeds in two phases. In the first or "programming" phase, the user specifies the system by providing:

- (1) A grammar specified in an augmented context-free production notation. Recursion is permitted. The augments consist of LISP-like commands for storage, retrieval, and comparison of information. Augments are executed when encountered during a parse. Syntax to the right of an augment will be pursued only when the execution yields a non-false (nil) value. Augments can be used to check semantic acceptability and to call for functional response actions.
- (2) A dictionary input file containing four items for each vocabulary word: dictionary spelling, English spelling, syntax class name (which serves to relate the dictionary to the grammar), and a 9-bit numerical field for arbitrary (usually semantic) information accessed by augments in the grammar. A word may have alternative dictionary spellings. In our application, phonological and front-end dependent rules are applied to nominal phonemic spellings to produce the dictionary spellings.

- (3) A scoring matrix which serves to relate the dictionary symbols (an arbitrary set) to the APEL string input. In work to date, we have used a scoring matrix derived from front-end experience with the corpus of 73 sentences.

To complete the first phase, CASPERS compiles the grammar, dictionary input, and scoring matrix into internal forms designed for efficient access during the second or "run time" phase.

Figure 14 shows a flow chart of the activities of CASPERS in its second or run-time phase, that is, as it transforms an APEL string into sentences. As the figure suggests, the transformation process is a cyclic one beginning (at the bottom of the figure) with path selection and ending with path updating. A path in CASPERS is a sequence of words which have been recognized by the DICTIONARY DEGARBLE and which, as far as the left-to-right parse has proceeded, could constitute a sentence compatible with the grammar. The PATH LIST is an ordered list of pointers to all paths which have been pursued during the parse. The paths are ordered according to the average score per input symbol (APEL) consumed. In addition to remembering the words making up a path, CASPERS retains information about the grammatical structure of each path and about word alternatives at each word position of each path which has not yet been pursued by the parsing process.

In operation, CASPERS selects the highest scoring path and uses the grammar to predict legal syntax classes for successor words to add to the selected path. It then searches the dictionary for all words which fall in the legal syntax classes and scores them with respect to the input APEL string. In general, more than one successor word is found, and the PATH LIST is updated by adding a new path formed by extending the previously selected path with the highest scoring successor word. (The path-selection process removed the originally selected path from the list, and the new path is placed on the list according to its score.)

In addition to adding the path found by extending the previously highest scoring path with a successor word, the update process will, in general, produce another new path for consideration by replacing the rightmost word of the previously highest scoring path with its next highest scoring alternative word. Thus, on each cycle the system can produce two new paths for evaluation on future cycles.

When a selected path corresponds to a complete sentence as specified by the grammar, the sentence is placed on the COMPLETION LIST. The user has the option of specifying that the system continue operating until any arbitrary number of sentence candidates have been found or until all possibilities have been exhausted. In our application, we also require that, to be acceptable, a sentence must consume essentially all the input APELs. This condition is controlled by one of the augments to the grammar.

In the process of matching dictionary words with input APELs, the DICTIONARY DEGARBLE must not only handle situations where symbol substitutions have occurred, but must deal successfully with cases where the input string does not contain the same number of symbols as the dictionary representation of the word. CASPERS attempts to handle this so-called "alignment" problem by scoring a single input string against a single dictionary spelling in more than one manner. The strategy is to allow a given input symbol to be scored in all three of the following ways: against the same dictionary unit as the preceding input symbol (known as a "merge" operation), against the next dictionary unit (the "normal" mode), and against both the next dictionary unit and its successor (known as a "splitting" operation). The example in Fig. 15 illustrates this approach to scoring.

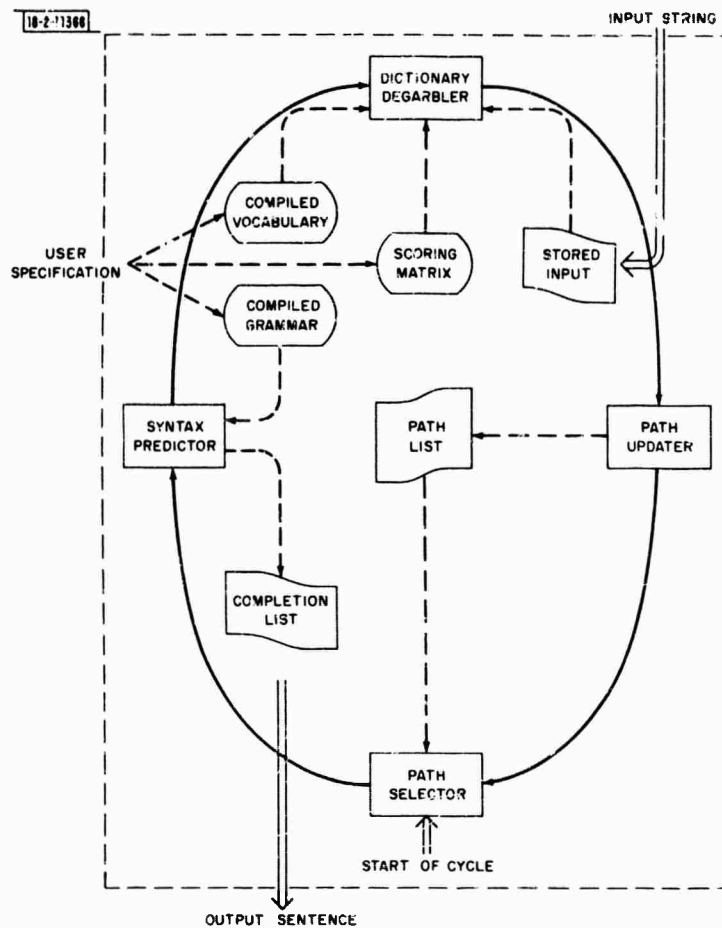


Fig. 14. Flow chart for CASPERS system operation.

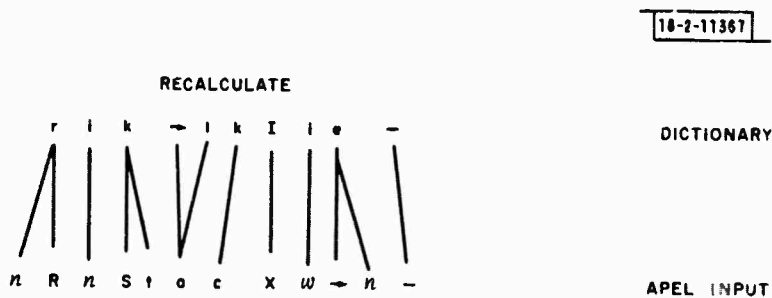


Fig. 15. Illustration of CASPERS merge-split dictionary degarbler strategy.

The final score for a successfully matching pattern is computed as the sum of the scores produced by the input symbols divided by the total number of input symbols consumed. It is quite possible for a given word to consume a varying number of input symbols.

Evidence to date indicates that this split-merge strategy is a reasonably effective means of handling the alignment problem. One might expect a prohibitively large number of alternative match patterns to be generated, but, in practice, it appears that the poor scores which many of the possibilities receive from the scoring matrix are effective in eliminating them at an early stage of the process.

The split-merge strategy, coupled with an arbitrary rule which retains the last APEL in a preceding word for matching with a current word, has shown some success in handling phonological effects at word boundaries. It is clear, however, that a more general approach to word boundary problems is desirable, and work is under way on an extension to CASPERS which will allow the user, when he provides his dictionary information, to specify a set of rules to govern phonetic transformations that can occur at word boundaries. This capability will allow CASPERS to handle phonological rules such as the one that states that a /d-y/ sequence (as in "did you") can be changed to the single phoneme /J/. Phonological rules which apply within words are handled with multiple pronunciations (rules applied at dictionary compile time) rather than rules to be tested at run time.

Two different linguistic modules using CASPERS have been produced and operated successfully with the acoustic phonetic front end. One module makes use of a vocabulary of about 200 words and a grammar of more than 175 productions. The other has a vocabulary of about 500 words and 300 productions. Results with the smaller vocabulary module have been encouraging. Implementation problems have limited testing of the larger vocabulary module. These problems have recently been overcome and both modules are now being actively tested and modified.

### C. General Syntax Program (GSP)

We are developing a third linguistic module based on the General Syntax Processor concept of R. Kaplan.<sup>4</sup> Our interest in GSP in addition to the VASSAL and CASPERS approaches derives from two properties of GSP. One is its ability to allow flexible control of parsing strategy by the user. In both VASSAL and CASPERS, parse strategy is largely locked into the system implementation and is, therefore, not readily varied. The other property is GSP's orientation toward parallel processing. We anticipate that specialized processor hardware will ultimately be required if linguistic processing is to be carried out at real-time rates and reasonable costs, and we expect that parallel-processing techniques will prove valuable in achieving the required performance. GSP is a good vehicle for exploring the application of parallel processing to parsing problems in our existing computer facility.

Most of our work to date on GSP has been concerned with implementation of the basic system with graphic displays and tablet interaction. The operation of GSP was described in some detail in the previous SATS.<sup>5</sup> To review briefly, GSP has the following basic components: (1) a grammar (which could be thought of as specifying instructions to be executed); (2) a chart (which could be thought of as the data on which the grammar operates); (3) a set of configurations, each of which specifies a part of the grammar to be applied to a part of the chart and also contains a list of registers representing tentative conclusions about structures; and (4) an algorithm for choosing which configuration from the set to try at a given point in the parsing (this being necessary since the TX-2 is not a parallel-processing machine). Definite conclusions about the structures are

represented in the chart in order to be available to other configurations. An important part of our implementation is a graphic representation of the chart with information about each configuration displayed over the part of the chart on which the configuration will act.

Work on GSP has been in three areas: (1) improving the information presented in the graphic display, (2) increasing the amount of user control via typed commands and tablet, and (3) making the control algorithm more flexible to allow the study of a range of orders of execution.

Our implementation of GSP gives the user extensive control over the order in which configurations are tried. The user exercises his control by setting the seven light buttons which appear at the top of the chart display, and which are described briefly in Table II below. The first three buttons determine when the program stops, and the last four are used by the program to

TABLE II THE ORDER CONTROL LIGHT BUTTONS		
State 1	State 2	Brief Description
proc-stop	proc-go	stop when process changes
path-stop	path-go	stop when path changes
arc-stop	arc-go	stop after each arc is executed
anyproc	samproc	try to make next configuration from same process
dosubr	~dosubr	execute configuration sending to port
docaller	~docaller	execute configuration waiting at port
depthfirst	breadthfirst	make basic order depth or breadth first

determine what configuration is to be executed next. When the program stops, the user then has a number of options available. He may ask to see the list of configurations in the order they will be tried next, he may ask that the program be resumed with the configuration that would have been used next, and he may change the configuration to be used next.

We have also provided the user with many controls (typically exercised by drawing a character over a particular box in the chart) to be of use when the chart will not fit on the scope. These include being able to shift the chart right or left, as well as up or down, and to eliminate the display of most of a particular box.

We have begun work on getting GSP to parse a sentence from APELs. As a preliminary step, we have integrated a word-finding process (which, however, will require considerable modification to run on APELs) with a syntax process. We have also begun work on a program to translate from a modified phrase structure grammar form (which is easier to write in) to the GSP form.

A paper entitled "The Use of Graphics in Viewing Linguistic Structures Produced by the General Syntax Processor" was presented at the annual meeting of the Association for Computational Linguistics in Ann Arbor, Michigan on 1 August 1973. The paper was accompanied by a movie showing the graphics in action.

#### IV. MID-TERM SYSTEM TASK

The task of the mid-term system is to allow vocal command of a system for studying the acoustic correlates of phonemic events. It uses the Lincoln speech data base, graphic, and various other facilities available on TX-2.

A researcher, by voice commands, searches the data base for instances of a class of phonemic events in a specified context. (This assumes that the utterances have been labeled.) He collects at least two measurements from the acoustic data for each selected event, tests the usefulness of pairs of measurements for identifying the events, and displays the results of these tests. He may also display the acoustic data associated with particular events. Thus, the goal of the system is to enable the researcher to gain insights about what measurements to use and how to use them to make phonemic identifications.

#### A. Subtasks

The system is organized into a number of subtasks. The following description of these subtasks gives a more detailed idea of what the researcher can make the system do for him.

##### 1. Specify the Phonemic Sample

###### a. Specify the Phonemes of Interest

The list of phonemes of interest to the user is called the "search list"; it is initially empty. It may take several utterances to get the desired phonemes into the search list. Phonemes can be removed from, as well as added to, the search list. An utterance that affects the search list may have one of several forms.

To a limited degree, a phoneme class can be specified according to some of the common articulatory terms. The user may indicate single phonetic classes such as vowels, nasals, labials. Such a class may be restricted by certain modifiers, e.g., high vowels, unvoiced labials. The command may contain a sequence of phoneme names. The phonemes are named using a naming convention that is similar to the "spelling" alphabet used in communication. Phonemes may be taken out of the search list by an "exclude" command, which has a form similar to those above.

###### b. Specify the Phonetic Context

The context, defined as one phoneme to the left and one phoneme to the right, consists of two lists of phonemes called "left context" and "right context." These lists are like the search list, initially empty. Phonemes are added to them as described above for the search list.

###### c. Additional Restrictions

The user may specify utterance initial and utterance final environments.

##### 2. Specify Conditions on Utterances from Which Events Will Be Taken

The user can restrict the utterances according to speaker, sex, site, and date of entry into the data base.

##### 3. Specify Measurements

The user may specify up to six measurements from existing fields in the data base. The measurement will be taken from a specified place in the event. Some simple calculations on measurements, mean, maximum, minimum, and sum are allowed.

#### 4. Examine the Samples

The user may look at various statistics concerning the context and measurement he has selected. For example, he may look at the distribution of events, or display means, standard deviations, or sample sizes for specific contexts.

#### 5. Run a Statistical Discrimination Program

The user may apply one of a set of models to any two of the measurements he has selected. The hand-produced labels are used to generate the appropriate statistics, and decision rules are generated. These rules may then be used to classify the events in the current sample or may be used with an independent sample. The models include weighted and unweighted Euclidean, generalized, and city block distance.

#### 6. Examine the Results

The user may look at a confusion matrix for his classification scheme, or he may plot his data on the scope in an attempt to discern which events tend to be misclassified.

#### 7. Use Display Programs

The user may examine in a larger context the data from which his measurements were taken. Displays of single valued functions of time, formants, and spectrograms are available. These programs might also be used in Subtask 3 where the user is trying to decide what measurements to use.

#### 8. Query and Change the System State

The user will want to verify that the search list or context has been correctly formed; or he may not remember how he defined a particular measurement. So, he can query the system state and, if it does not agree with his expectations, he may modify it.

### B. User Model

There is some constraint on the order in which subtasks are done. For example, the user cannot examine the results before specifying the phonemes of interest. But the first three subtasks (i.e., specifying the phonemic sample, utterance conditions, and measurements) can be done in any order. Subtask 4 might be omitted (examining the samples). Even Subtask 5 (run discrimination program) can be omitted, as there will be a default discriminant program that would be run when the user attempted to do Subtask 6 (examine results) if the results did not yet exist. Subtasks 5 and 6 might be done several times in succession with different discriminant programs specified in Subtask 5. Subtasks 7 and 8 (display and query or change system state) might be done at any time.

It should be noted that the actual search of the data base to obtain the sample is not done until some subtask is attempted which requires it, which could be Subtasks 4, 5, or 6. Thus, the system keeps track of its state, and what actually gets done for a subtask depends on the system state.

It should be clear from the above that, although the task is goal directed, the user is not constrained to follow a straight-line path through the subtasks. We expect that his path will include a number of loops. This is particularly true since the user will be able to log out of the system with his work in progress, so that he can, at a later time, log in again and continue it.

The result of this is that there will probably be very few cases where the meaning of an utterance is greatly restricted because of the subtasks that have been done already or have not yet been done.

### C. Grammar

The grammar for this task was constructed by a study of the subtasks, and enumerating the various ways in which the subtasks could be expressed. There are a number of general restrictions that the user must be aware of:

- (1) Utterances must be "complete" and clearly demarcated by silence from succeeding utterances without the use of conjunctions or vocal gestures.
- (2) An utterance contains information about only one subtask at a time. Thus, one cannot say "I want to study intervocalic nasals for male speakers."
- (3) All utterances are effectively commands. Therefore, "I want to identify fricatives" is, in effect, "Identify fricatives" and "What's in the left context" is "Show the left context." Thus, a number of verb phrases are equivalent to a simple command verb.
- (4) Judgments have been made about the plausibility of what are permissible syntactic constraints. Thus, "labial fricatives" is permissible, but not "fricative labials." It is interesting that even in this limited domain, the plausibility of an utterance can be difficult to decide.

Three parsing systems have been built and each has its own internal representation of the grammar. Externally, the grammar is represented in BNF. The nonterminal classes are semantic rather than syntactic categories. For example, words denoting the phonemes are put into a nonterminal class called ph. Externally, the grammar makes no explicit distinction of syntactic categories such as noun phrase, verb, adjective, etc.

The dictionary has columns for the phonetic spelling, the orthographic spelling, and semantic class.

### D. Functional Response

When the linguistic module successfully processes an utterance, control passes to the functional response module. The orthographic transcription of the utterance and a calling sequence indicating which functional response has been requested and the arguments of the response are made available by the linguistic module.

At this point, it is perhaps clearer to consider three components of the functional response module (they are not necessarily distinct from a program organization viewpoint).

The first component must consult the task model, determine whether what has been requested seems appropriate, and, if so, update the task model and ensure that appropriate action follows. For example, suppose that the linguistic module indicates that the phonemes /i/ and /I/ are to be excluded from the search context. It must be determined whether this is an appropriate action at this stage of the investigation.

The second component is a collection of subroutines that actually perform the desired actions. In the above example, the search list would be examined to see if /i/ and /I/ actually appeared there, and, if so, these phonemes would be deleted. A large number of the action routines already exist. These are simply the data base storage, retrieval, and display programs which have been operational for over a year.

While all the above are well within the range of modern user programming, the third component has received too little attention but is fundamental to a speech understanding system. This component deals with the apparent inconsistency of a user's request with respect to the task/user model or the data base. The major problem, of course, is to determine why the system is in this inconsistent state. It may be because of a logical bug in the programming itself, and we clearly recognize this possibility and our responsibility to minimize such situations. On the other hand, it may be that certain arguments derived from the utterance by the linguistic module do not reflect what the speaker actually said. While semantic constraints can do much to eliminate this type of problem, it is not always realistic to do this at the linguistic processing stage. For example, suppose it is determined that the user apparently requested that his work from a prior session be retrieved from the data base entry '160'. When the data base is searched, it is discovered that the only entries are numbered from 1 to 90. Our general system philosophy of separation of functions and minimum response time to clear and correct requests compels us to separate information of this detail from the normal operation of the linguistic module. When the apparent inconsistency arises, we must make the information available so that the linguistic module can determine if perhaps the utterance was other than first indicated. In the final analysis, despite any ingenuity that we might bring to bear, there will be instances in which we will have to admit that either we do not understand the request or that the user has asked us to do something impossible. In this case, we can only relay to the user what it is we thought he said, why we cannot do it, and ask him to reconsider and/or rephrase his request.

## V. SPEECH DATA BASE

The speech data base continues to provide essential support at all levels to work on a speech understanding system. At present, hand labels, the results of acoustic phonetic processing, formants, and the volume function are available on-line for about 185 utterances. Spectra, pitch, etc., are on tape. With the new disk, we are now able to increase the on-line data.

The data prepared for the July workshop at Carnegie-Mellon University provided a realistic comparison of segmentation efforts. One result of this workshop was an agreement not to pre-emphasize the 20-kHz waveform. Any data entered into the data base as of September 1973 have the 10-kHz waveform pre-emphasized as described in the previous SATS. The 20-kHz waveform is not pre-emphasized.

An 8-minute 16-mm demonstration movie was made which shows how a speech analyst might use the Lincoln Speech Data Facility (LSDF) to investigate a problem of current interest. A narrator, following a prepared script, explains the organization of the LSDF and the actions of the analyst as he proceeds through the problem. The movie was shown at the eleventh Annual Meeting of the Association for Computational Linguistics held at the University of Michigan on 1-2 August 1973.

### A. Data Base Software

The extensive software that was built to support the speech effort has proved its effectiveness as the amount of speech work on TX-2 has greatly increased. Daily use of the data base by the acoustic-phonetic researchers led to the design and implementation of two improvements in the user interface.

The major improvement was the specification of a standard 9-character utterance identifier comprising the following elements: "batch code" (i.e., code for the recording session in which

the utterance was made), site code, speaker code, sentence-list code, and sentence number. An index associates the identifier for each utterance with an indirect reference to its drum location.

The use of the identifier was extended by the implementation of an "asterisk convention" which allows the simple specification of certain common sets of utterances. If asterisks are used in place of any element, it is understood that any value for that element is acceptable. Thus, the identifier "03\*\*\*01\*\*" stands for the set of all utterances in batch 3 from sentence list 1, regardless of speaker or sentence number. A control program was written that accepts a set specification and a program name and applies the program to each utterance in the set.

A second improvement in the user interface was the implementation of programs for converting the internal form of event arrays into character files, and the reverse. This allows the scope editor to be used for inputting the results of hand labeling while retaining the original internal form for efficient program use.

Further improvements were made to the speech data display system. These allow hard copy of a broad range of data. A multi-label facility was implemented so that any number of labelings can be compared easily.

## B. SURNET

The SURNET server was designed initially to meet the specialized needs of the speech research community. It was expected that various sites would implement SURNET user processes fitted to these specialized needs. However, since many sites already had general-purpose file transfer protocol (FTP) user programs in operation, it was decided that SURNET should be converted to an FTP server rather than have the sites add to their programming load. This has now been done. The new server has been written and is now being tested.

At present, the server will respond to a limited command set (USER, RETR BYE) and meet the minimum declarative specifications. The user can retrieve data from the speech data base. He cannot store data. At present, the user cannot retrieve TX-2 (i.e., APEX) files.

## VI. SYSTEM ACTIVITIES

### A. TX-2 System Hardware

The new IBM 3830/3330 disk system has reached operational status. Addition of this disk system approximately doubles the direct access storage capacity of TX-2. Its 17-msec rotation time and 806-kbytes/sec transfer rate offer performance 3 to 5 times faster than the Unimac FASTRAND II drum system which has supported the APEX time-sharing system for almost ten years.

To support the new disk, the TX-2 I/O channel hardware was modified to allow independent operation on its own memory port, and a channel interface equivalent to an IBM Block Multiplexer was constructed.

In addition to the new disk, a new tape system was installed during this reporting period. The new system offers higher performance at lower cost, and has resulted in higher reliability as well. TX-2 now has two 9-track 800/1600 bpi, 125 ips drives and one 7-track 556/800 bpi drive which is used primarily for communication with other computers.

The control units of both the tape and disk systems are capable of running sophisticated off-line diagnostics provided by the manufacturers. Such capability is a great help in connecting

devices to one-of-a-kind computers such as TX-2. In our case, it was only necessary to write new diagnostic programs for the disk interface.

#### B. TX-2 System Software

The new disk memory system has been incorporated into the APEX time-sharing system and is now being used for medium term storage of files and time-sharing swapping of users. All public and private user files have been transferred from the Univac FASTRAND 11 drum to the disk. The vacated drum space is now available for use by the speech data base.

To support use of the disk for APEX files, an allocation scheme was designed and implemented for dynamically assigning and reclaiming disk space. APEX files are handled as units of 1 to 32 pages of 256 words each. The disk contains 411 cylinders of 19 tracks each. By formatting a track into no more than two records, we can store 10 pages of file information (including a few words of header information) on each track. In our allocation scheme, tracks are dynamically formatted as the need arises into either one 10-page record or two records whose sizes add up to 10 pages - 1-9, 2-8, 3-7, 4-6, or 5-5. When the records on a track are returned to free storage, the track is considered unformatted and may later be reformatted to satisfy future needs.

When a file is to be written on the disk, an attempt is made to allocate it in the cylinder at which the read-write heads are then located. If successful, delay due to head movement is avoided. Also, if a file requires more than one record, all records must be in the same cylinder. This too avoids head movement during the reading or writing of the file, and also minimizes the number of bits required for the disk address of the file.

Features of the allocation scheme increase the likelihood of accommodating a file in a given cylinder. For each file size, the allocation program has a table of combinations of record sizes which will accommodate the file. The entries in the table are arranged in a priority fashion with the highest priority given to combinations which (1) have minimal latency wastage between records, (2) utilize record sizes frequently encountered, and (3) leave as residues useful free record sizes. When these goals are contradictory, a compromise is used.

As an example, consider the case of a 32-page file. Such a file can be accommodated into the following combinations of record sizes, among others - 10-10-10-2, 9-9-9-5, 8-8-8-8, 10-9-8-5. A study of file size distribution in APEX reveals that 1-, 2-, and 32-page files are disproportionately common. Allocation of a 1-page file of necessity leaves a 9-page free residue on the track. Similarly, a 2-pager leaves an 8 free. Therefore, highest priority for allocating 32-page files (and other sizes as well) is given to combinations which utilize the 8- and 9-page records which are likely to be common. If it is necessary to format one or more new tracks to accommodate a 32-page file, highest priority is given to 1-9 and 2-8 formats in order to provide free records for the common small files. In this way, allocating a file is an exercise in using what is available and providing useful residues for future requests.

Simulation studies of this scheme have shown that disk space utilization is quite high with fairly low need to move the read-write heads to accommodate a file. In one study, we filled a simulated disk with files whose sizes fit the usual APEX file size distribution.

Using the TX-2 hardware random-number generator, we chose a random file from this simulated disk, reclaimed its space, and then reallocated it in a randomly selected cylinder. The results were that about 52 percent of the files could be accommodated on the randomly selected cylinder, and about 28 percent could be accommodated on an adjacent cylinder.

### C. TSP System

TSP hardware reliability has improved to the point where most days are entirely free of errors. The TSP continues to be used to provide additional TX-2 terminals and to support some graphics work on the Laboratory's IBM 360/67.

The TSP is also being used for reading ARPA network mail from the BBN TENEX system. TENEX is most conveniently used in a full-duplex, character-at-a-time mode. The TSP was modified to allow the user to specify this mode as an option. Also, for the convenience of the mail readers, a system was implemented which allows a user to get a hard copy of the text displayed on his terminal from the LDX printer attached to the TX-2.

On 1 January 1974, a new TELNET protocol will become mandatory on the ARPA network. The TSP system has already been modified so that it will work acceptably with this new protocol. Among other things, the new protocol will change the way in which a user specifies the full-duplex system. To retain full-duplex capability with TENEX, we will have to modify the TSP system when TENEX begins to conform to the new protocol.

### REFERENCES

1. Speech Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (30 November 1972), pp. 1-6, DDC AD-754940; and Speech Understanding Systems Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (31 May 1973), pp. 1-6, DDC AD-763723.
2. G. E. Peterson and H. L. Barney, "Control Methods Used in a Study of the Vowels," *J. Acoust. Soc. Am.* 24, 175-184 (1952).
3. L. J. Gerstman, "Classification of Self-Normalized Vowels," *IEEE Trans. Audio Electroacoust.* AU-16, 78-80 (1968).
4. R. Kaplan, "A General Syntactic Processor," in *Natural Language Processing*, R. Rustin, Ed. (Algorithmics Press, New York, 1973), pp. 193-241.
5. Speech Understanding Systems Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (31 May 1973), pp. 12-15, DDC AD-763723.